CENG 3420
Computer Organization & Design

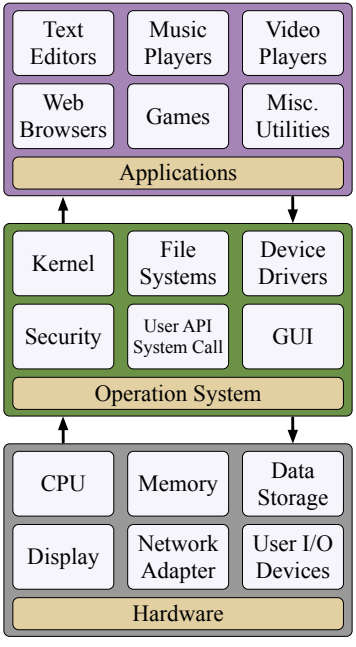
Lecture 03: ISA Introduction

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

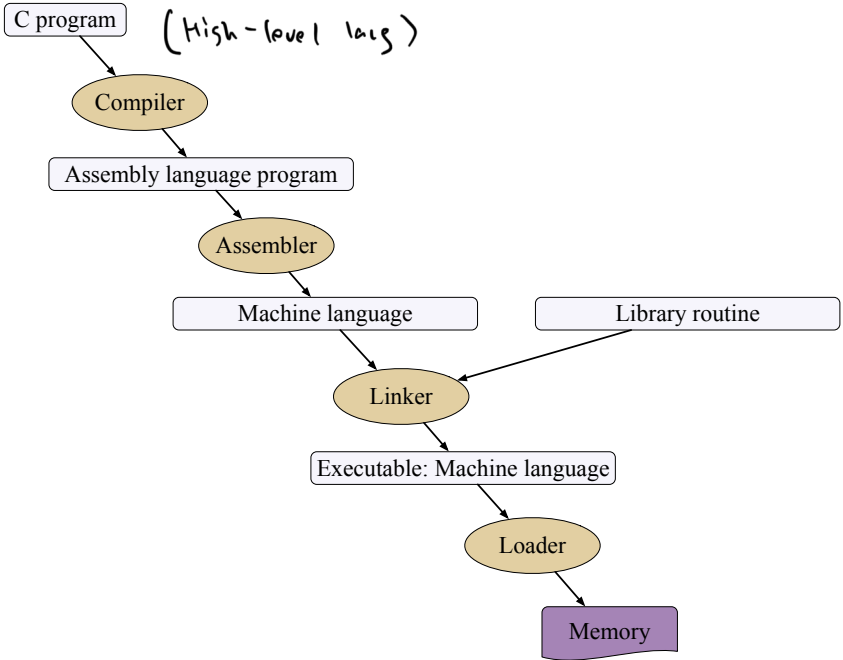
(Latest update: January 20, 2022)

Spring 2022

Computer System



Traditional Compilation Flow



Below the Program

- High-level language program (in C)

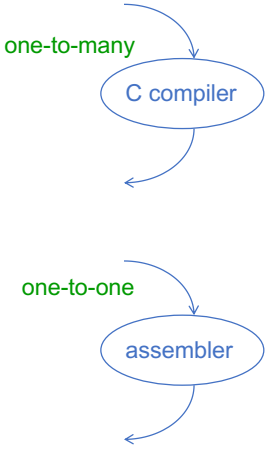
```
swap (int v[], int k)
(int temp;
       temp = v[k];
       v[k] = v[k+1];
       v[k+1] = temp;
)
```

- Assembly language program

```
swap:  sll    $2, $5, 2
       add    $2, $4, $2
       lw     $15, 0($2)
       lw     $16, 4($2)
       sw     $16, 0($2)
       sw     $15, 4($2)
       jr     $31
```

- Machine (object) code

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
. . .
```



Below the Program

- High-level language program (in C)

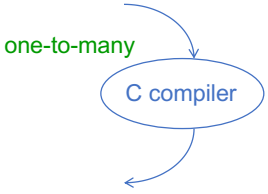
```
swap (int v[], int k)
(int temp;
       temp = v[k];
       v[k] = v[k+1];
       v[k+1] = temp;
)
```

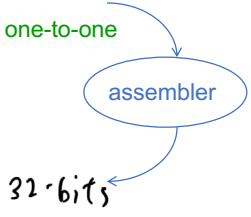
- Assembly language program

```
swap:  sll    $2, $5, 2
       add    $2, $4, $2
       lw     $15, 0($2)
       lw     $16, 4($2)
       sw     $16, 0($2)
       sw     $15, 4($2)
       jr     $31
```

- Machine (object) code

```
000000 00000 00101 00010000100000000
000000 00100 00010 00010000000100000
. . .
```

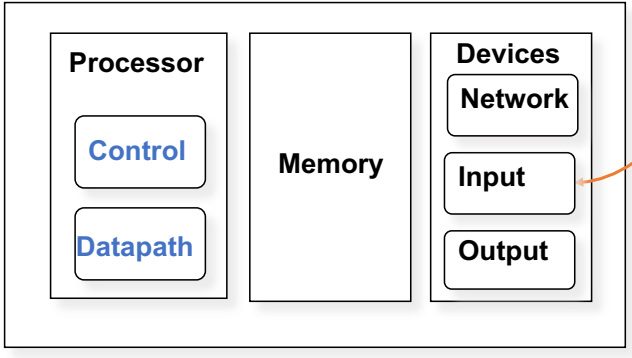




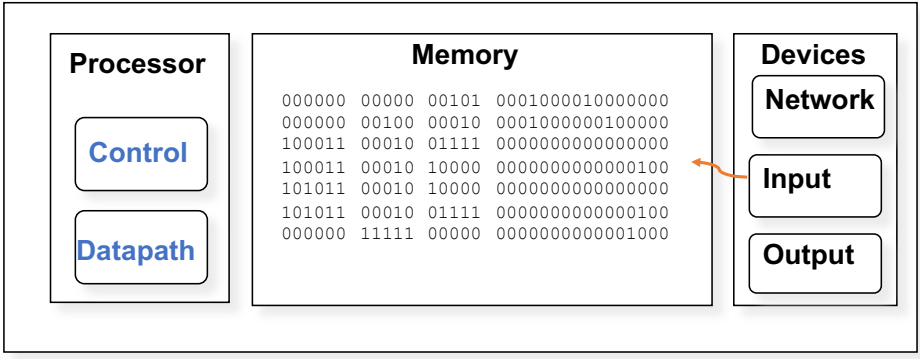
Max # of operations?

Input Device Inputs Object Code

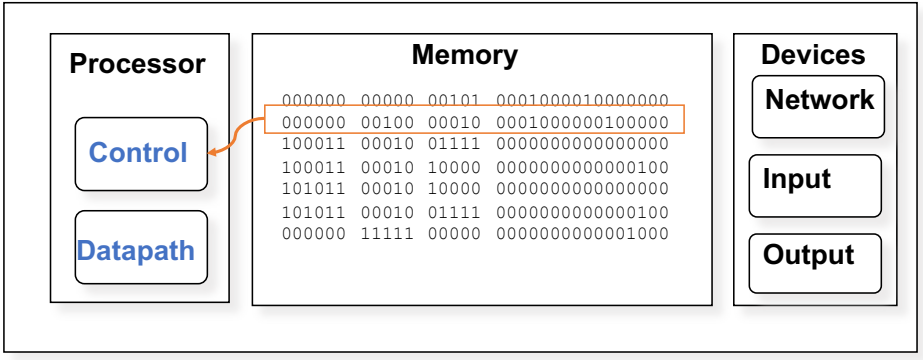
```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
100011 00010 01111 0000000000000000
100011 00010 10000 0000000000000100
101011 00010 10000 0000000000000000
101011 00010 01111 0000000000000100
000000 11111 00000 0000000000001000
```



Object Code Stored in Memory

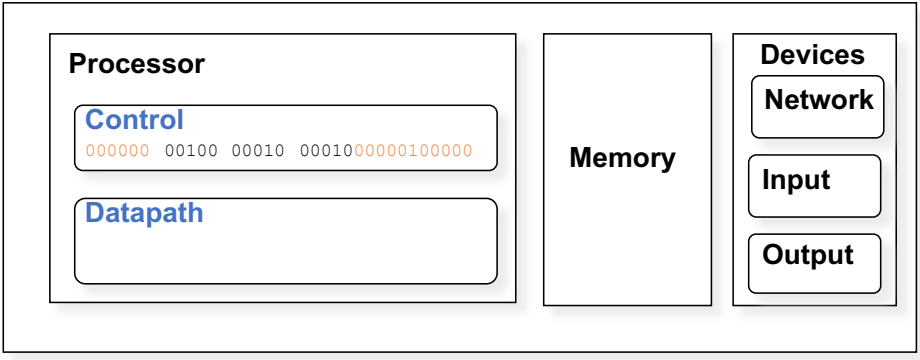


Object Code Stored in Memory



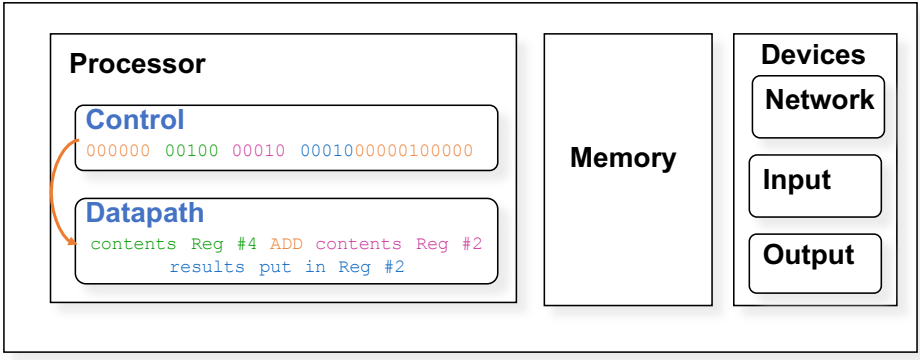
Processor fetches an instruction from memory

Decode & Excute Codes



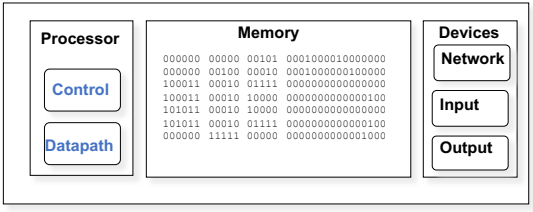
- Control decodes the instruction to determine what to execute

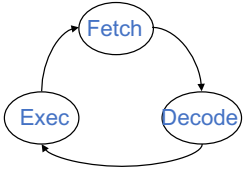
Decode & Excute Codes



- Control decodes the instruction to determine what to execute
- Datapath executes the instruction as directed by control

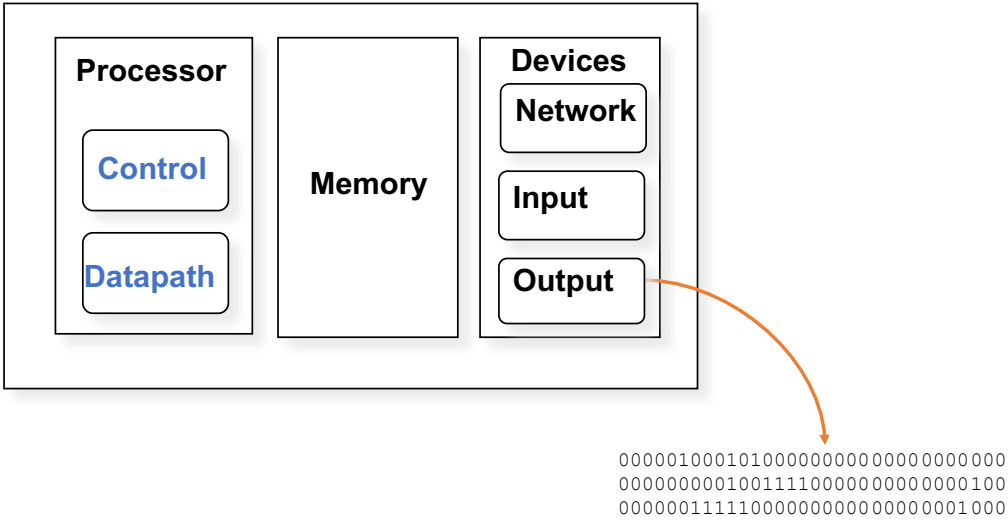
What Happens Next?



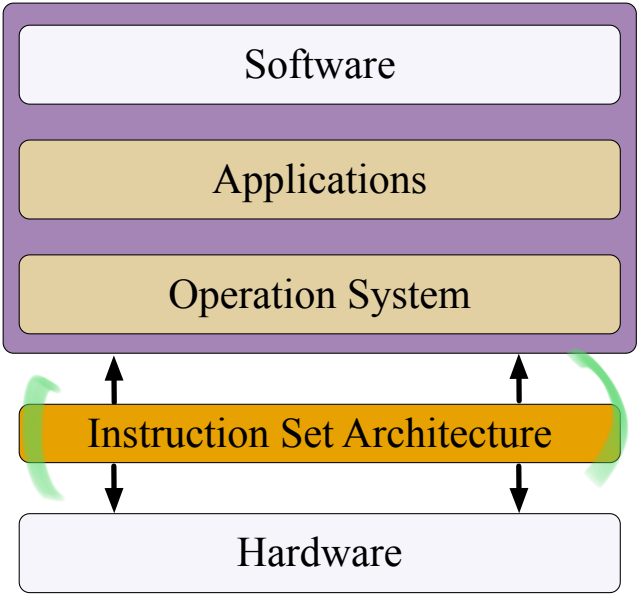


- Processor fetches the next instruction from memory
- How does it know which location in memory to fetch from next?

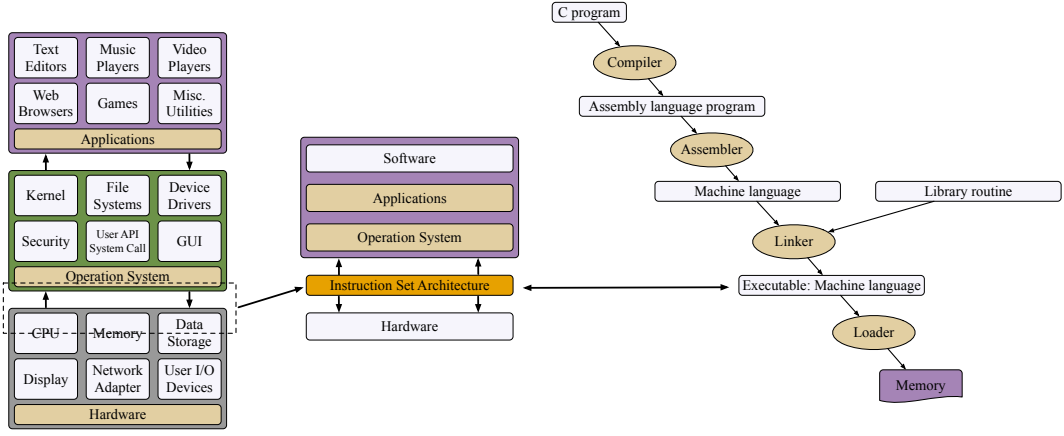
Output Device Outputs Data



ISA – Bridge between Hardware & Software

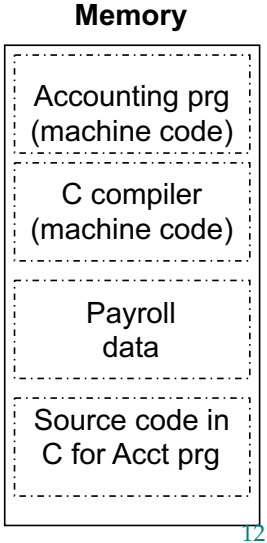


Connection


Text Editors
Music Players
Video Players
Web Browsers
Games
Misc. Utilities
Applications
Kernel
File Systems
Device Drivers
Security
User API System Call
GUI
Operation System
CPU
Memory
Data Storage
Display
Network Adapter
User I/O Devices
Hardware
Software
Applications
Operation System
Instruction Set Architecture
Hardware
C program
Compiler
Assembly language program
Assembler
Machine language
Library routine
Linker
Executable: Machine language
Loader
Memory

Two Key Principles of Machine Design

1. Instructions are represented as numbers and, as such, are indistinguishable from data
2. Programs are stored in alterable memory (that can be read or written to) just like data



Stored-Program Concept

- Programs can be shipped as files of binary numbers – binary compatibility
- Computers can inherit ready-made software provided they are compatible with an existing ISA – leads industry to align around a small number of ISAs

The language of the machine

- Want an ISA that makes it easy to build the hardware and the compiler while maximizing performance and minimizing cost

Our target: the RISC-V ISA

- similar to other ISAs developed since the 1980's
- RISC-V is originated from MIPS, the latter of which is used by Broadcom, Cisco, NEC, Nintendo, Sony, ...

Design Goals

Maximize performance, minimize cost, reduce design time (time-to-market), minimize memory space (embedded systems), minimize power consumption (mobile systems)

CISC vs. RISC

Complex Instruction Set Computer (CISC)

Lots of instructions of variable size, very memory optimal, typically less registers.

- Intel x86

Reduced Instruction Set Computer (RISC)

Instructions, all of a fixed size, more registers, optimized for speed. Usually called a "Load/Store" architecture.

- RISC-V, LC-3b, MIPS, Sun SPARC, HP PA-RISC, IBM PowerPC ...

History of MIPS (cont.)

- Used in many embedded systems
- E.g., Nintendo-64, Playstation 1, Playstation 2

RISC – Reduced Instruction Set Computer

RISC Philosophy

- fixed instruction lengths
- load-store instruction sets
- limited number of addressing modes
- limited number of operations

- Instruction sets are measured by how well compilers use them as opposed to how well assembly language programmers use them

RISC-V (RISC) Design Principles

Simplicity favors regularity

- fixed size instructions
- small number of instruction formats
- opcode always the first 6 bits

Smaller is faster

- limited instruction set
- limited number of registers in register file
- limited number of addressing modes

Make the common case fast

- arithmetic operands from the register file (load-store machine)
- allow instructions to contain immediate operands

Good design demands good compromises

- For RV32I, 4 base instruction formats (R/I/S/U) and 2 extended instruction formats (B/J)